TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Modul pro podporu procesorů Vinculum na existujícím výukovém přípravku

Module with processor Vinculum for existing education kit

Bakalářská práce

Autor: **Daniel Vích**

Vedoucí práce: Ing. Tomáš Martinec, Ph.D.

Konzultant: Ing. Josef Grosman

V Liberci 16.5.2012

Fakulta mechatroniky, informatiky a mezioborových studií

Zadání bakalářské práce

Příjmení a jméno studenta (osobní číslo - nepovinné)	Vích Daniel M09000193
Zkratka pracoviště	MTI
Datum zadání BP/DP	1.10.2011
Plánované datum odevzdání	20.5.2012
Rozsah grafických prací	dle potřeby dokumentace
Rozsah průvodní zprávy	cca 30 – 40 stran
Název BP/DP (česky)	Modul pro podporu procesorů Vinculum na existujícím výukovém přípravku
Název BP/DP (anglicky)	Module with processor Vinculum for existing educational kit

Zásady pro vypracování BP/DP (text nijak neformátujte, pouze očíslujte jednotlivé body .. 1) ... 2) ... atd. a každý bod uveďte jako nový odstavec):

1, Seznamte se s obvody Vinculum a jejich možnostmi
2, Navrhněte rozšíření výukového přípravku o podporu těchto obvodů tak, aby bylo možné využít optimálně všechny periférie na přípravku
3, Realizujte navržené rozšíření, otestujte ho ve spojení s přípravkem a připravte sadu příkladů pro jeho využití při výuce

Seznam odborné literatury (text nijak neformátujte, pouze každou položku uveďte jako nový odstavec):

[1] Dokumentace k obvodům Vinculum
[2] Dokumentace k vývojovému přípravku

Vedoucí BP/DP	Ing. Tomáš Martinec, Ph.D.
Konzultant BP/DP (u externích pracovníků uveďte plný název pracoviště – firmy)	Ing. Josef Grosman

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych rád poděkoval vedoucímu mé bakalářské práce panu Ing, Tomáši Martincovi, Ph.D. za rady, připomínky a čas, který mé práci věnoval. Dále bych chtěl také poděkovat svým rodičům a celé rodině za podporu při studiu.

Abstrakt

Tato bakalářská práce se zabývá popisem mikroprocesoru Vinculum od firmy FTDI s podporou USB Host integrovanou přímo v čipu. Dále také návrhem a sestavením modulu s tímto mikroprocesorem na již existující univerzální výukový přípravek a vytvoření třech úloh použitelných při výuce. Tyto úlohy jsou vytvářeny nad operačním systémem dodávaným výrobcem, který je v této práci také popisován. Zájemce by měl tak získat základní informace o využití tohoto mikroprocesoru, způsobu návrhu zapojení a programovaní aplikací na tomto obvodu.

Abstract

The bachelor work is concerned with description of the microprocessor from the FTDI Vinculum with USB Host support directly integrated in the chip. It also design and module constructing with the microprocessor to the existing universal development board for education and the creation of three tasks applicable to teaching. These tasks are created by on the special operating system supplied by the manufacturer, which in this work is also described. Those interested should obtain the basic information on the use of microprocessor design method of wiring and programming of applications for this circuit.

Klíčová slova: modul s mikroprocesorem Vinculum, VNC2, VOS, integrovaný obvod, výukový přípravek

Keywords: Vinculum microprocessor module, VNC2, VOS, integrated circuit, teaching development board

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých termínů

USB – Univerzální sériová sběrnice, která zajišťuje připojení periferií nebo přenos dat.

Harvardská architektura - Architektura mikroprocesoru, která fyzicky odděluje paměť programu a dat a jejich spojovací obvody.

DMA - Způsob přímé komunikace hardwarového subsystému s operační pamětí. To znamená, že procesor při této komunikaci může vykonávat jinou činnost.

CISC - Anglicky *Complex instruction set komputer.* Označuje sadu strojových instrukcí procesoru, kde je pokryto široké množství instrukcí, některé by šli naprogramovat i pomocí jednodušších instrukcí.

Flash paměť - Je nevolatilní (semipermanentní) elektricky programovatelná (zapisovatelná) paměť s libovolným přístupem.

UART – Zařízení pro sériovou komunikaci.

SPI - Je sériové periferní rozhraní. Používá se pro komunikaci mezi řídícími mikroprocesory a ostatními integrovanými obvody.

LQFP – Z angličtiny low profile quad flat package, je to typ pouzdra integrovaného obvodu.

Pull-up rezistor – Chrání vstup oproti nežádoucím stavům při stavu vysoké impedance. Snaží se udržet logickou úroveň 1.

Pull-down rezistor – Je opakem pull-up rezistoru. Snaží se udržet logickou úroveň 0.

Schedulling - Je proces rozhodování o tom, jak rozdělit zdroje mezi několik úkolů.

Mikrojádro – Jádro operačního systému, které je velmi malé a používá pouze základní funkce.

BOMS – Třída USB mass storage device, která slouží k ukládání dat na paměťová média.

FAT – Označuje souborový systém používaný na paměťových médiích

Vlákno – Označuje odlehčený proces (program) přizpůsobený codu multitaskingu. Vlákna mezi sebou sdílejí společný paměťový prostor, ale i další struktury.

Semafor – Je synchronizační primitivum, který se používá zejména k ochraně proti souběhu.

1 Úvod

V dnešní době se stále více ve většině typech různých vestavných zařízení jak v domácnosti tak i v průmyslových aplikacích používá moderní rozhranní USB. Které slouží například k přenosu dat z a do zařízení nebo i k připojení dalších doplňujících periferii jako jsou klávesnice, webové kamery, tiskárny, čtečky čárových kódů a spoustu dalších. Tyto zařízení ve svém nitru skrývají mikroprocesor, který se stará právě o řízení práce s vyjmenovanými prvky. Proto je vhodné, aby každý programátor těchto mikroprocesorů věděl, jak se s tímto rozhranním USB pracuje.

Cílem této práce je seznámení se s jedním z řady mikroprocesorů, kteří mají právě podporu rozhranní USB již v sobě integrovanou. Vytvoření hardwarového modulu a jeho realizace s tímto procesorem na již existující výukový přípravek, který slouží k výuce předmětu Počítačový hardware a rozhranní na Technické univerzitě v Liberci. A vytvoření několika příkladů pro využití při výuce, které budou poukazovat na základní práci s tímto mikroprocesorem, využití možností operačního systému mikroprocesoru a základní komunikaci s USB zařízením, konkrétně s USB flash diskem.

2 Popis mikroprocesoru Vinculum

Mikroprocesor se nyní vyrábí ve 2 verzích a to VNC1L a VNC2.Zásadní rozdíl mezi těmito dvěma verzemi je v architektuře, oba dva mají Harvardskou architekturu, ale VNC1L má 8 bitovou oproti VNC2, který má 16 bitovou. Jako další rozdíl mezi těmito typy jsou velikostí pamětí VNC1L má paměť RAM 4kB a FLASH 64kB, zatímco VNC2 disponuje RAM o velikosti 16kB a FLASH 256kB. Další vlastnosti se liší jen v malých detailech, takže jsou takřka totožné, proto je budu popisovat pro obě verze společně a případné rozdíly zdůrazním.

Jako nejlepší vlastnost těchto mikroprocesorů je samozřejmě podpora USB. Na obou verzích se nachází dvě full-speed nebo low-speed rozhraní s podporou Host/Slave (Master/slave), kde host slouží právě k připojení různých externích zařízení a Slave umožňuje připojení k jinému Host zařízení (např. PC). Každý z obou portů může být nastaven samostatně, takže např. 1. bude Host a 2. Slave. Pro komunikaci mezi USB a externí sběrnicí I/O jsou integrovány dva DMA řadiče, které umožňují hardwarovou akceleraci přenosu dat, takže procesor může při tomto přenosu vykonávat jinou činnost.

Jako další rozhraní uvedu UART, SPI a paralelní FIFO. Je tu ale znovu rozdíl mezi VNC1L a VNC2 a to v tom, že VNC1L má pouze SPI slave, zatímco VNC2 má SPI master i 2x slave.

2.1 Elektrické vlastnosti

- Napájecí napětí obvodu V_{cc} je 0 do 3.6V.
- Napětí na vstupech -0.5 do +5.00V.
- Ztrátový výkon (při V_{cc} = 3.6V) 250mW
- Provozní napájecí proud 25mA
- Napájecí proud v režimu Suspend od 1 do 2mA

2.2 Vstupně/výstupní porty (I/O)

Jak VNC1L tak VNC2 disponují porty pro připojení periferií (čidel, snímačů, maticové klávesnice,…).

2.2.1 VNC1L

Má těchto portů 28 a všechny mají pevně dané, na kterém pinu se nacházejí.

2.2.2 VNC2

Počet I/O portů liší podle pouzdra:

- 44 I/O portů na 64 pinovém pouzdře
- 28 I/O portů na 48 pinovém pouzdře
- 12 I/O portů na 32 pinovém pouzdře

U všech typů pouzder je použit I/O multiplexer, kterým si můžeme určit, na kterém pinu bude která periferie. To umožňuje jednoduší návrh pro případné aplikace. Zde je seznam periferií, které si pomocí multiplexoru můžeme rozmístit na jednotlivé piny:

- UART
- SPI slave0
- SPI slave 1
- SPI master
- FIFO – asynchronní
- FIFO – synchronní
- GPIO - všeobecně použitelné piny
- Debug rozhranní
- PWM

Tabulka 1 udává maximální počet pinů požadovaných pro jednotlivé periferie. V závislosti na konstrukci nemusí být použito všech 9 pinů rozhraní UART a to samé platí i pro GPIO.

Periferie	Maximální požadovaný počet pinů
UART	9
SPI slave 0	4
SPI slave 1	4
SPI master	5
FIFO asynchronní	12
FIFO synchronní	14
GPIO	40
Debug	1
PWM	8

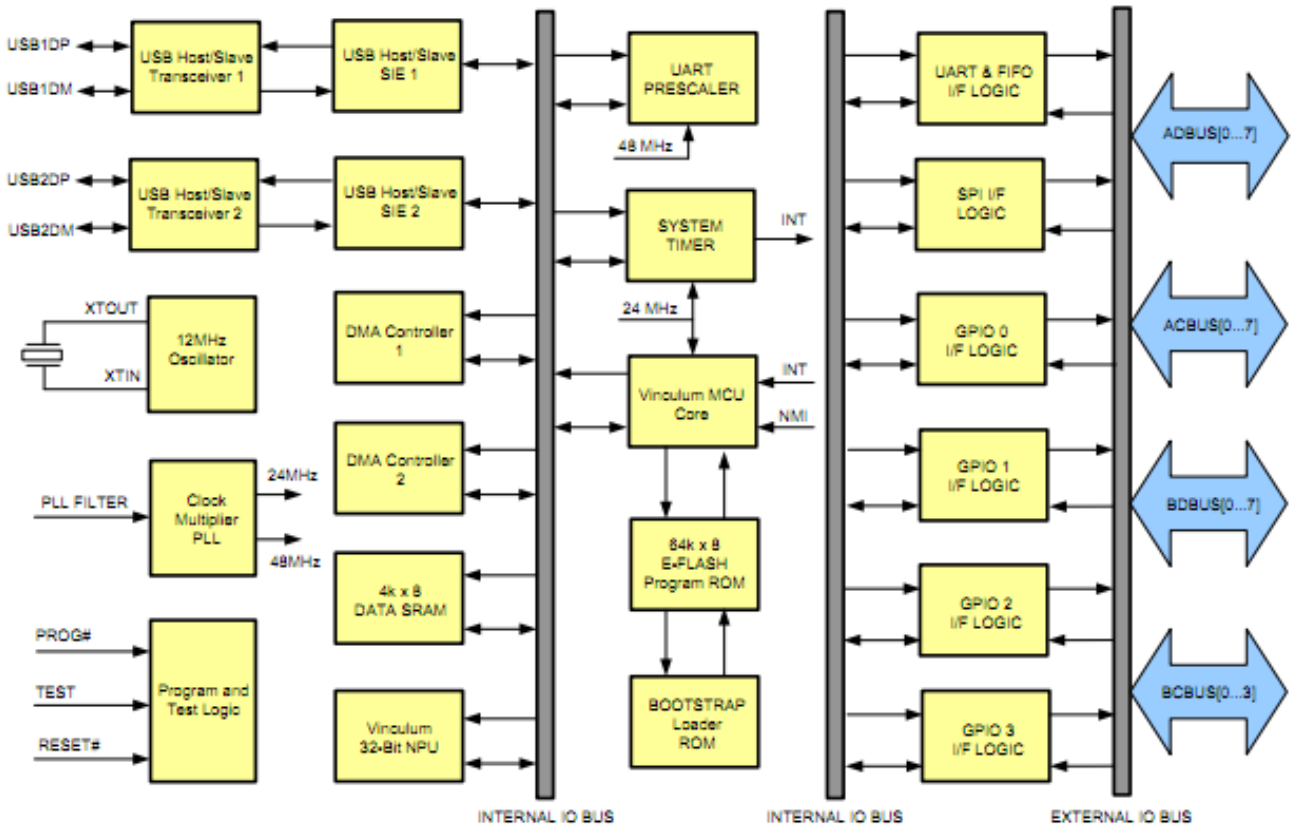
Tabulka 1 - Maximalní počet periferií na piny

2.3 Mechanické vlastnosti

- VNC1L se vyrábí v pouzdře se 48 piny splňující standard LQFP (Low-profile quad flat package).
- VNC2 v těchto typech pouzder:
    - 32 pinů standardu LQFP (Low-profile quad flat package)
    - 32 pinů standardu QFN (Quad flat no Leeds)
    - 48 pinů standardu LQFP (Low-profile quad flat package)
    - 48 pinů standardu QFN (Quad flat no Leeds)
    - 64 pinů standardu LQFP (Low-profile quad flat package)
    - 64 pinů standardu QFN (Quad flat no Leeds)

2.4 Blokové diagramy

2.4.1 VNC1L



Obrázek 1 - blokový diagram VNC1L

Popis jednotlivých bloků

Vinculum MCU Core: 8bitové jádro procesoru s harvardskou architekturou, to znamená, že má oddělenou paměť dat a programu. Používá rozšířenou sadu instrukcí CISC technologie.

System timer: zajišťuje přerušení pro VNC1L firmware

E-FLASH program ROM: VNC1L má k dispozici 64kB Flash paměti programu. Mikroprocesor samostatně se dodává nenaprogramovaný, takže vyžaduje před použitím naprogramování přes bootstrap loader.

Bootstrap loader ROM: Je to malý blok nesmazatelné paměti (4kB), ve kterém je bootloader, který mimojiné umožňuje programování přes rozhranní UART v případě mikroprocesoru bez firmwaru. Po nahrání firmwaru (jakýkoliv od výrobce) lze pak programovat i přes rozhranní USB.

NPU (numeric co-processor): tento blok umožňuje až 32bitové výpočty. Důležité uplatnění má v implementaci souborového systému FAT (zejména při adresaci).

UART prescaler: zajišťuje hodiny pro UART blok. Může se změnit jeho hodnota, podle které se mění přenosová rychlost na rozhranní UART a to v rozsahu 300Baud až 1MBaud.

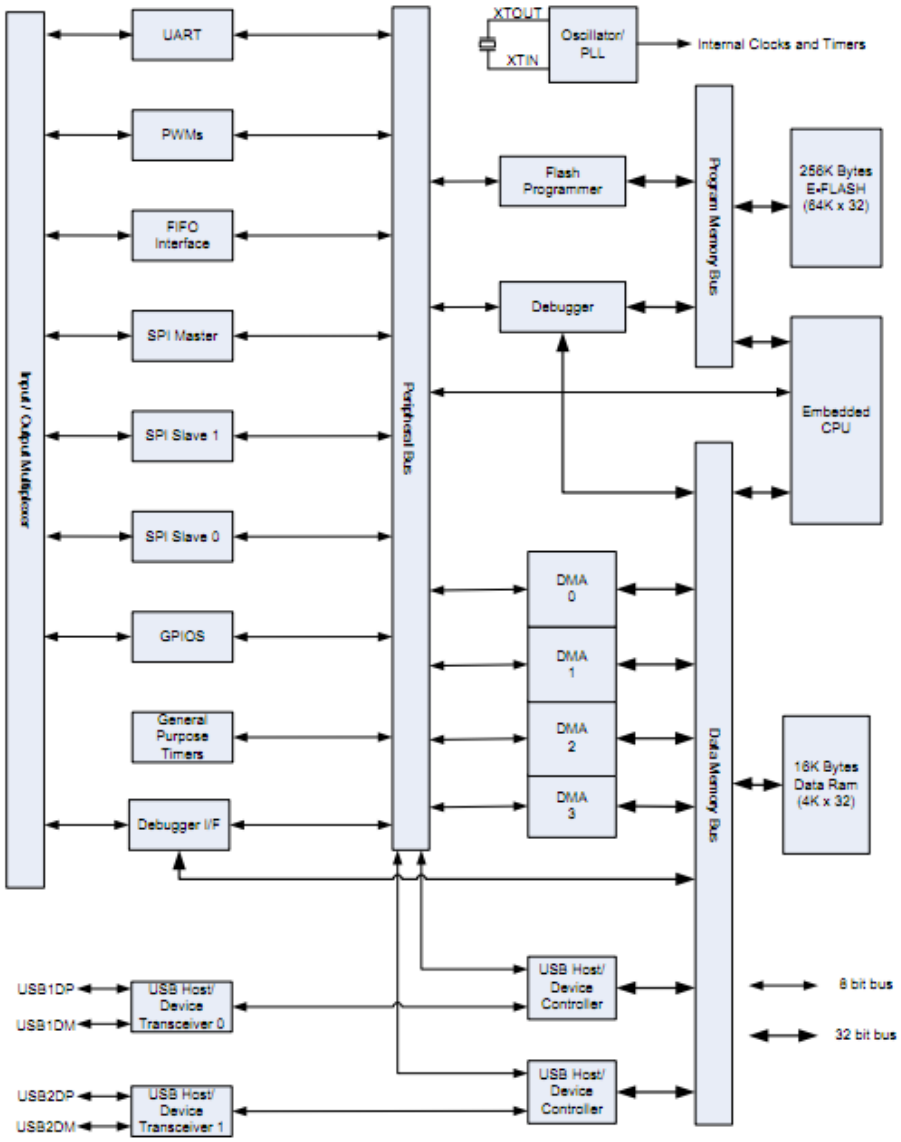
12 MHz Oscillator: generuje hodinový signál z externího 12 MHz krystalu

Clock multiplier PLL: jeho vstupem jsou 12 MHz hodiny a generuje 24MHz a 48MHz hodinový signál pro jádro mikroprocesoru, UART prescaller, system timer a USB SIE bloky.

USB serial interface engine (SIE): zajišťují fyzickou komunikaci po sběrnici USB, včetně uplatnění protokolů pro kontrolu chyb.

DMA controller: výrazně zvyšuje výkon mikroprocesoru tím, že data z rozhraní USB SIE, UART, SPI a FIFO převádí mezi kterékoliv jiné prostřednictvím datové SRAM s minimální účastí jádra. Příkladem použití může být zobrazování obrazu ze zapojené webkamery v USB na displej zapojený například na rozhranní SPI.

2.4.2 VNC2



Obrázek 2 - blokový diagram VNC2

Popis jednotlivých bloků

V této části popíšu pouze odlišné vlastnosti jednotlivých modulů oproti VNC1L.

Embedded CPU: 16bitové jádro procesoru s harvardskou architekturou, to znamená, že má oddělenou paměť dat a programu. Používá rozšířenou sadu instrukcí CISC technologie.

Input/Output multiplexer: umožňuje při návrhu výběr portu na kterém se daná periferie bude nacházet.

2.5 Úspora energie

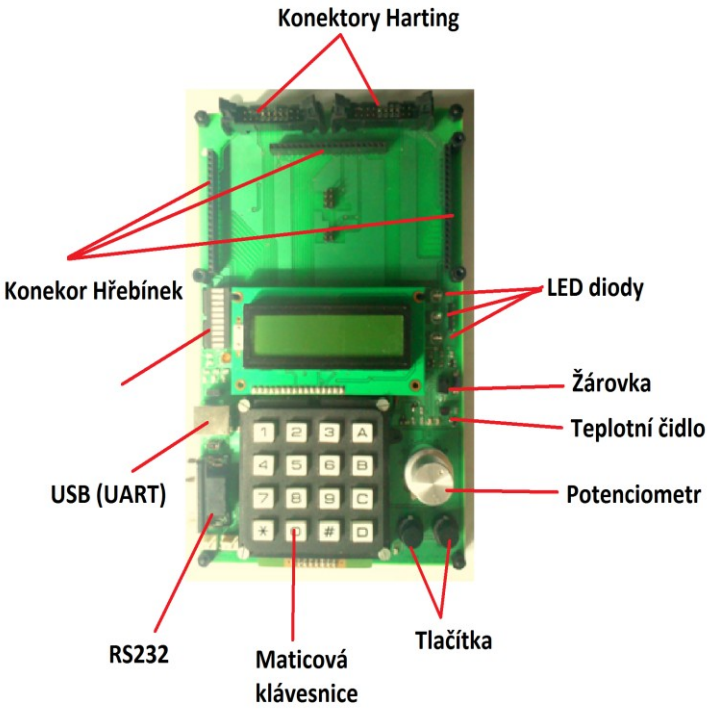
Pro aplikace, kde napájení obvodu bude řešeno pomocí baterie je potřeba co největší úspora energie. Verze VNC2, má pro tuto aplikaci k dispozici 3 režimy úspory energie jedná se o snížení operační frekvence. Nejnižší 12MHz, střední 24MHz a běžná frekvence 48MHz. Toto je k dispozici při použití originálního firmwaru s použitím Vinculum operačního systému nebo jinak RTOS (real time operating systém).

VNC2 může být i v tzv. Standby módu, ten musí být implementován ve firmwaru. V tomto módu je plně vypnutý hodinový signál a napájení některých funkčních bloků. Pro probuzení z tohoto módu musí přijít některý z těchto signálů: USB0/1 DP nebo DM, SPI slave 0 select (spi_s0_ss#), SPI slave 1select(spi_s1_ss#) nebo UART ring indicator (uart_ri#).

3 Výukovový přípravek – univerzální kit

Výukový přípravek je v podstatě základní deska osazená několika základními periferiemi, které jsou vyvedeny do 2 typů konektorů pro zapojení modulu s procesorem. 1. typem konektoru je klasický 2,54mm hřebínek a 2. Konektor Harting. Co se týče mechanického zpracování tohoto kitu jedná se o dvouvrstvou desku plošných spojů s kombinací montáží THT a SMD. Periferie na tomto kitu jsou následující:

- Dvouřádkový LCD display
- Maticová klávesnice
- 2 tlačítka
- Potenciometr
- 3x LED dioda
- LED bar
- Mini žárovka
- Piezo reproduktor
- Procesor reálného času DS1338C-33
- Rozhranní RS-232
- Rozhranní RS-485/RS-422
- Rozhranní USB UART – FTDI FT232RL



Obrázek 3 - výukový přípravek

Tento univerzální přípravek byl vytvořen panem Ing. Tomášem Martincem, Ph.D. a slouží také k výuce předmětu počítačový hardware a rozhranní na Technické univerzitě v Liberci. Existuje na něj již několik modulů s mikroprocesory, zatím ale bez podpory rozhranní USB host.

4 Modul s procesorem Vinculum

Tento modul je jedním z výsledků této práce a obsahuje mikroprocesor Vinculum, konkrétně typ VNC2-64L1B. Osazuje se na výše zmíněný univerzální kit a obstarává veškerou komunikaci s periferiemi na univerzálním kitu.

4.1 Návrh zapojení

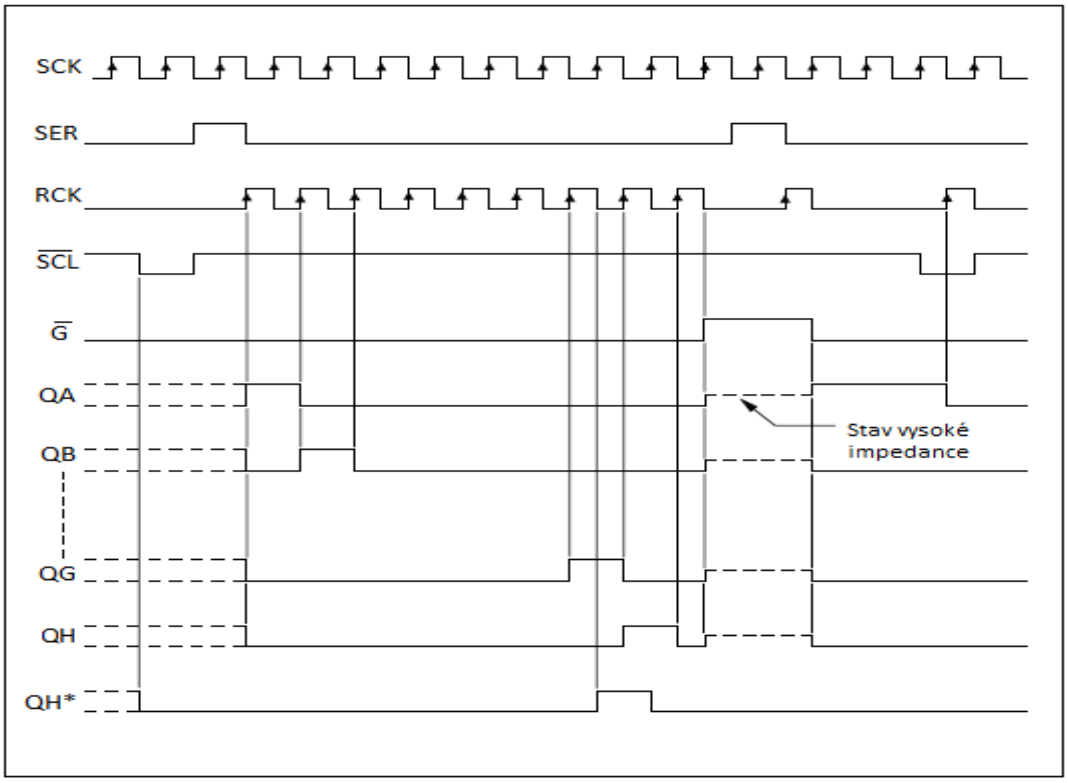
Samotný návrh zapojení se odvíjel od univerzálního kitu, kde nejdůležitějším parametrem byla volba typu procesoru, tak aby byly využity všechny periferie. Nabízeli se verze s 28 I/O porty nebo 44 I/O porty konkrétně se jednalo o typy procesoru VNC2-48L1A a VNC2-64L1B. Každá z těchto verzí má vyvedeny dva konektory USB a použit stejný obvod pro ovládání ledbaru. Tento obvod se skládá ze dvou posuvných registrů 74HC595D. Ostatní, zejména pasivní součástky, se pak od jednotlivých verzí trochu liší.

Napájení celého zapojení je zajištěno z univerzálního kitu, kde jsou k dispozici napěťové úrovně 5V a 3,3V. Kde napětí 3,3V je zajištěno z regulátoru LM3940 a je jím napájen mikroprocesor. Tyto napětí jsou vyvedena na konektoru J1, J2, J3 znázorněné na schématu.

Jednotlivé návrhy zapojení jsem vytvořil v softwaru EAGLE od společnosti CadSoft, který mi usnadnil práci s tímto návrhem, zejména s kontrolou zapojení. Návrh v tomto softwaru se vytváří vkládáním součástek do schématu a jejich spojování vodiči. Program disponuje velkou knihovnou součástek, ale v některých případech jsem si součástky musel vytvářet sám, jak jejich schematickou značku, tak i pouzdro, které pak sloužilo pro návrh desky plošných spojů. Jednalo se zejména o samotný mikroprocesor a použité konektory.

4.1.1 Posuvný registr 74HC595D

Je 8 bitový sériový posuvný registr s ukládacím registrem a 3stavovým výstupem. Posuvný a ukládací registr mají na sobě nezávislý hodinový signál. Na vstup SER vstupují data, při přivedení jedničkového signálu na vstup SCK nastává posun dat. Z jednotlivých registrů, při vstupu jedničkového signálu na vstup RCK, se data přesunou do ukládacího registru. Pokud jsou oba vstupy SCK a RCK napojeny na stejný hodinový signál data se přesouvají okamžitě na ukládací registr.



Obrázek 4 – stavový diagram posuvného registru 74HC595D

Tento obvod je vybaven asynchronním resetem SCL pro všech 8 stupňů posuvného registru, aktivní je při hodnotě 0. Hodnoty na ukládacím registru se objeví na výstupech Q v případě, když je negovaný vstup G nastaven na hodnotu 0. Výstup QH* je sériový a slouží ke kaskádování posuvných registrů.

4.1.2 Návrh zapojení s 28 I/O porty

První návrh s 28 porty vynikal využitím všech I/O, ale ne všechny periferie byly připojeny k procesoru napřímo. Jednalo se o 6 I/O portů o které se dělily periferie ledbar, tlačítko 2, LED dioda 2, piezo reproduktor, teploměr, potenciometr a procesor reálného času. Výběr těchto periferii zprostředkovávala volba jumperů na jednotlivých I/O portech. Pro ovládání ledbaru je potřeba 5 pinů a procesoru reálného času 2 pinů. Zbylé periferie jsou po 1 pinu. Ke komunikaci v tomto návrhu slouží piny mikroproceosru 31 – 34, kde rozmístění komunikačních signálů je následující:

- Pin 31 TXD
- Pin 32 RXD
- Pin 33 RTS
- Pin 34 CTS

Jelikož jsou na univerzálním kitu 2 možnosti připojení komunikačního rozhranní, tak na modulu lze pomocí jumperů JP3 a JP4 toto rozhranní přepnout. Takže si můžeme vybrat, zda použijeme rozhranní USB (UART) a nebo RS232.

Seznam použitých součástek

- Mikroprocesor VNC2-48L1A
- 2x posuvný registr 74HC595D
- Kondenzátory
  - 22pF – 2ks
  - 47pF – 4ks
  - 100nF – 5ks
  - 100uF – 2ks
  - 120 uF – 2ks
  - 10 nF -1ks
- Feritové jádro 600R@100MHz – 2ks
- Krystal 12MHz
- Rezistory
  - 47kΩ - 2ks
  - 27Ω -4ks
  - Odporová síť 8x56kΩ
- Konektor USB 2ks
- Konektory hřebínek

4.2 Použitý návrh zapojení s 44 I/O porty

Jelikož se první návrh nezdál příliš vhodný a nabízela se ještě verze mikroprocesoru se 44I/O porty vytvořil jsem tedy ještě tento druhý návrh, který byl nakonec použit pro další postup práce.

Při tomto návrhu byl použit mikroprocesor VNC2-64L1B, kde i při zapojení všech periferií jak na modulu, tak i na univerzálním kitu nebyli využity všechny piny. Nabízelo se tedy vyvést tyto piny do zvláštních konektorů pro případné další rozšíření.

Ke komunikaci s mikroprocesorem v tomto návrhu slouží piny 39 – 42 pro rozhranní UART, kde rozmístění jednotlivých signálů je následující:

- Pin 39 TXD
- Pin 40 RXD
- Pin 41 RTS
- Pin 42 CTS

Jelikož jsou na univerzálním kitu dvě možnosti připojení rozhranní, tak na modulu lze pomocí jumperů JP3 a JP4 toto rozhranní přepnout. Takže si můžeme opět vybrat, zda použijeme rozhranní USB (UART) a nebo RS232. Oproti předešlému návrhu je zde přidáno ještě debug rozhranní, které se nachází na pinu 11 a to nám pak dává další možnost jak tento mikroprocesor naprogramovat a zároveň nám umožňuje lepší odladění programu při použití debuggingu.

Po porovnání obou návrhů je tento druhý vhodnější k reálnému využití modulu. Důvodem této volby bylo, že nemůžeme najednou používat všechny periferie a také nepraktičnost při přepínání periferií, jelikož přepínání v předchozím návrhu zajišťovala volba jumperu. Dále chybělo rozhranní debug a také nemožnost rozšíření o další periferie. Jelikož je tento návrh vhodnější, další vývoj pokračoval na této verzi. Jednalo se zejména o rozmístění zapojení periferií, tak aby byli v souladu s návrhem desky plošných spojů. Schéma tohoto zapojení je umístěno v příloze.

Seznam použitých součástek

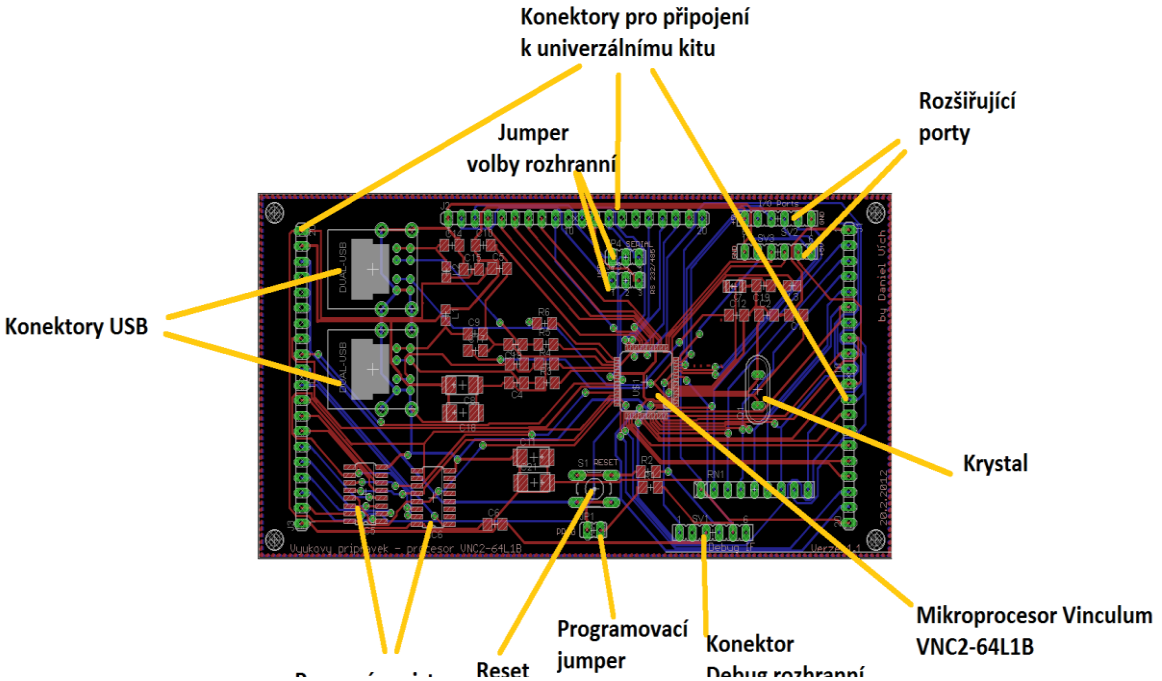
- Mikroprocesor VNC2-64L1B
- 2x posuvný registr 74HC595D
- Kondenzátory
    - 22pF – 2ks
    - 47pF – 4ks
    - 100nF – 6ks
    - 100uF – 2ks
    - 150 uF – 2ks
    - 10 nF -2ks
    - 4,7 uF – 1ks
- Feritové jádro 600R@100MHz – 3ks
- Krystal 12MHz
- Rezistory
    - 47kΩ - 2ks
    - 27Ω -4ks
    - Odporová síť 8x56kΩ
- Konektor Dual-USB 2ks
- Konektory hřebínek
- Konektor DUBOX
- Tlačítko reset

4.3 Optimalizace schématu a návrh desky plošných spojů

I když schéma obvodu obsahovalo již vše potřebné pro správnou funkci modulu, musela se provést optimalizace umístění periferií na jednotlivých pinech mikroprocesoru, tak aby pak při návrhu desky plošných spojů došlo k co nejmenšímu křížení vodičů. Hlavním parametrem při tomto rozvržení periferií na piny mikroprocesoru je velikost desky a rozmístění konektorů pro připojení k univerzálnímu kitu. Dále bylo důležité, aby na desce plošných spojů byli co nejkratší cesty datových vodičů a také šířka cest vodičů musela vyhovovat jejich elektrickému zatížení.

Rozměry desky plošných spojů byli pevně dány a určeny dle univerzálního kitu, konkrétně velikost udávalo rozmístění konektorů pro připojení modulu a vzdálenosti distančních sloupků pro pevné uchycení. Tyto parametry tedy udávali délku 120mm a šířku 60mm. Podstatné bylo také rozmístění konektorů USB vyvedených na modulu. V průběhu návrhu jsem uvažoval pouze jednoduchý konektor USB umístěný v krajích v horní části modulu, ale naskytl se problém s dostupností konektoru v případě zapojení konektoru Harting na univerzálním kitu. Musel jsem tedy zvolit jinou možnost umístění, v úvahu

připadalo vložení těchto konektorů nalevo nebo napravo na kraj modulu. Dolní část nepřipadala v úvahu z důvodu špatné dostupnosti ke konektoru, protože při vložení modulu do univerzálního kitu tomuto konektoru vadil LCD display a tedy by způsoboval špatnou manipulaci. Ve finální verzi bylo zvoleno umístění konektoru nalevo, ale i to ještě muselo podlehnout další úpravě, konkrétně se vyměnil jednoduchý USB konektor za dvojitý, kde pro připojení periferií bude sloužit pouze vrchní zdířka. Tato výměna jednoduchého konektoru za dvojitý měla za důsledek riziko možného zkratu nízkého konektoru, kvůli těsné blízkosti zasunutého konektoru periferie a vystouplých kontaktů hřebínkového konektoru. Výsledné rozmístění je vidět na Obrázek 5 - schéma rozmístění.



Obrázek 5 - schéma rozmístění

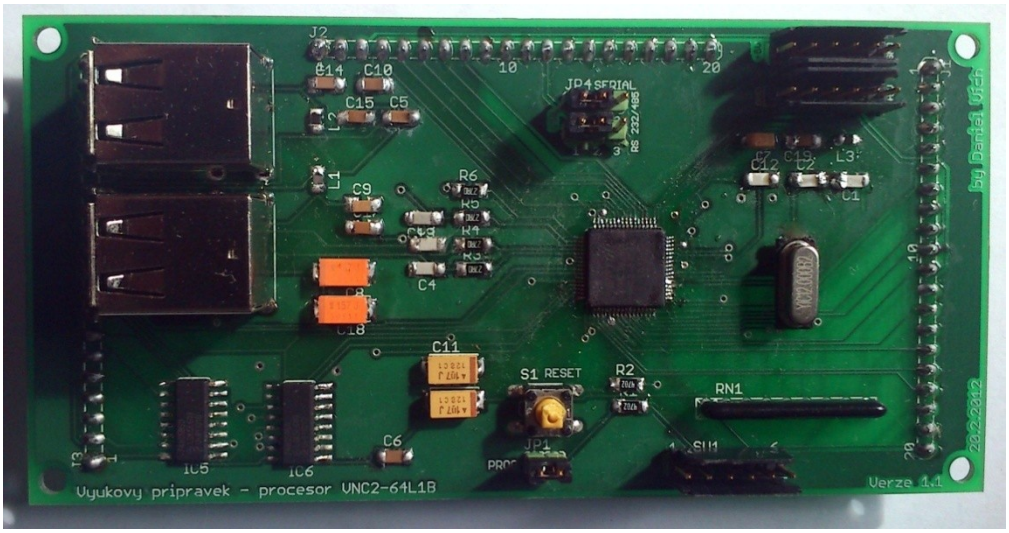
4.4 Výroba desky plošných spojů a osazení součástkami

Po dokončení všech úprav a kontroly návrhu desky plošných spojů bylo vše připravené pro výrobu. Všechny podklady pro výrobu se automaticky po nastavení formátu dat vygenerovaly z programu Eagle, ve kterém se prováděl i návrh. Tyto podklady představovali tzv. GERBER data ve správném formátu, konkrétně byl použit formát RS274-X. Tyto data reprezentovali jednotlivé vrstvy:

- Top – pomědění vrchní části desky
- Bottom – pomědění spodní části desky
- Top maska – nepájivá vrchní maska
- Bottom maska - nepájivá spodní maska
- Top potisk – vrchní potisk desky
- Bottom potisk- spodní potisk desky
- Obrys- obrys pro vyříznutí desky

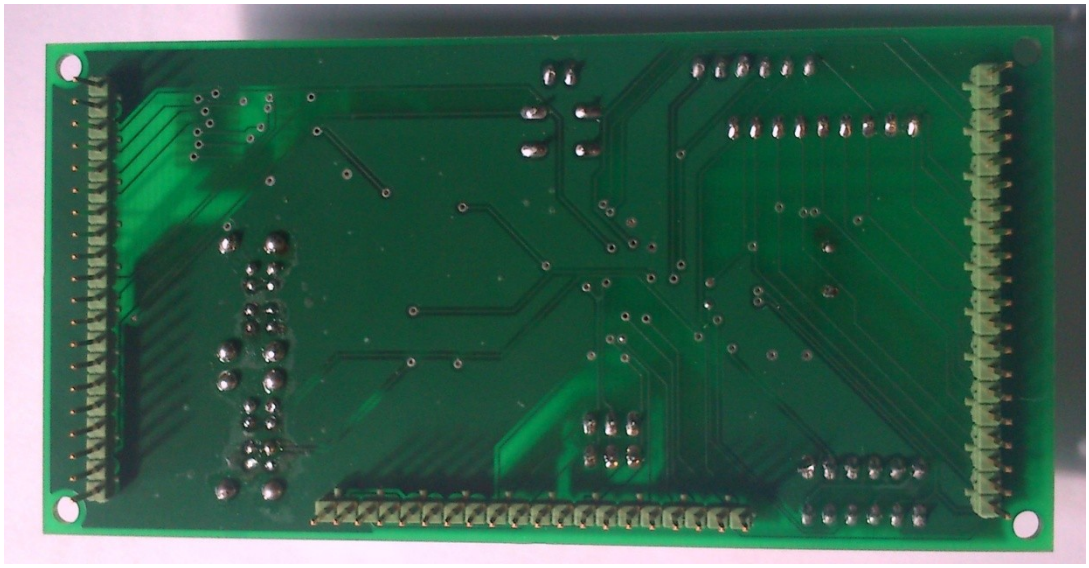
Jelikož se v návrhu vyskytují díry a pokovené propojky mezi vrstvami, bylo nutné také vygenerovat soubor pro vrtačku, kde se použil formát Excellon. Všechna tato data byla odeslána do POOL servisu, který zajišťuje společnost PragoBoard s.r.o., kde tato služba umožňuje levnější výrobu prototypů, protože podklady pro výrobu jsou jednorázové.

Po tom, co už byla deska plošných spojů zhotovena zbývalo pouhé osazení součástkami. Deska osazená součástkami je zobrazena na Obrázek 6 - osazená deska



Obrázek 6 - osazená deska plošných spojů ze strany součástek

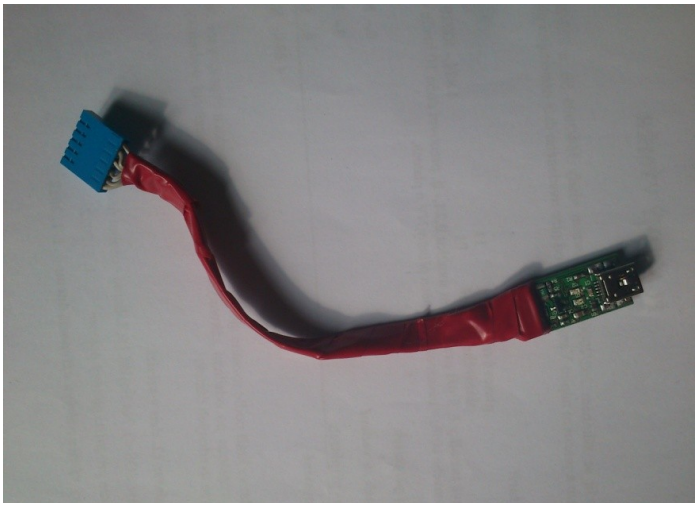
plošných spojů a Obrázek 7 - osazená deska plošných spojů ze strany konektoru k univerzálnímu kitu.



Obrázek 7 - osazená deska plošných spojů ze strany konektoru k univerzálnímu kitu

4.5 Debug rozhranní

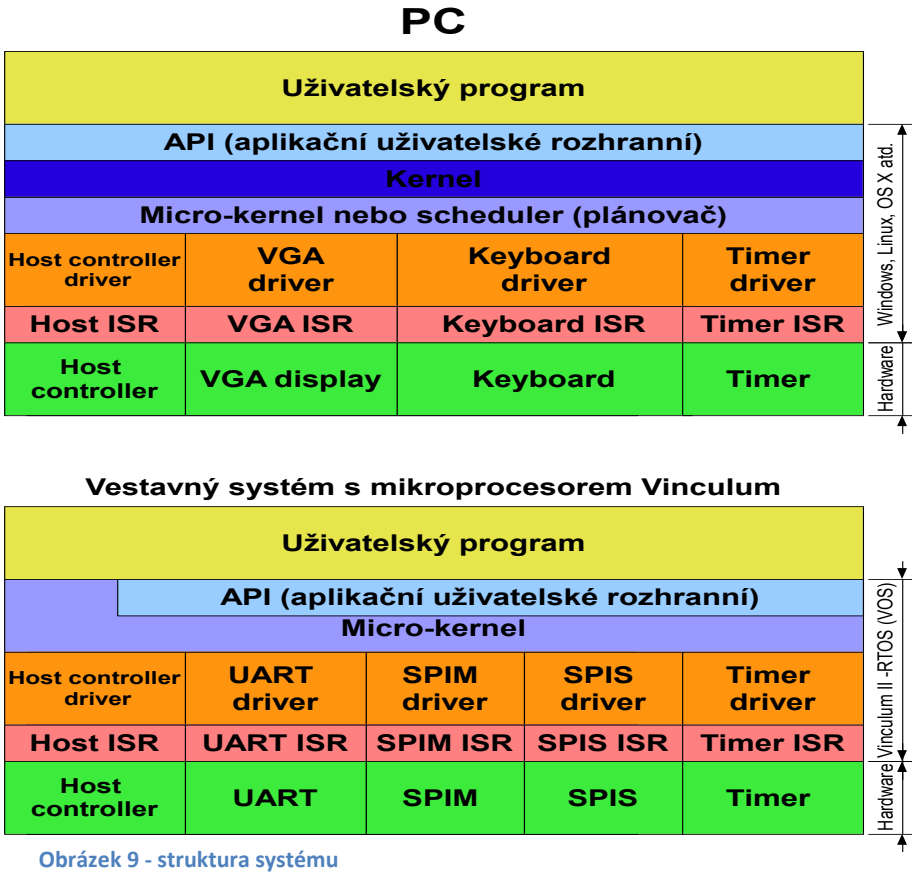
Debug rozhranní slouží k naprogramování a ladění obvodu. Pro použití tohoto rozhranní je potřeba speciální modul, který obsahuje upravený převaděč rozhranní USB – serial UART FTDI FT232R. Tento modul je pak propojen s počítačem přes USB a s modulem přes debug konektor, kde pak v počítači při použití vývojového prostředí od FTDI můžeme ladit aplikace s modulem.



Obrázek 8 - debug rozhranní

5 Programování obvodu

Programování mikroprocesoru Vinculum se díky implementaci mnoha rozhranní včetně USB může zdát složité, ale výrobce toto usnadnil svým vlastním API (aplikačním uživatelským rozhranním). Takže to znamená, že veškerý fyzický hardware překrývá operační systém, kde už máme hotové metody pro komunikaci a řízení hardwaru a právě přes tyto metody k hardwaru přistupujeme. S touto metodologií se můžeme běžně setkat na různých typech PC (osobních počítačích), kde je reprezentována operačními systémemy například Windows, Linux, OS X a dalších. U mikroprocesoru Vinculum tento operační systém pojmenovali jako Vinculum II RTOS (realtime operating system) nebo častěji používaný zkrácený název VOS (Vinculum operating system). Jedná se o operační systém s preemptivní metodou schedullingu. Na Obrázek 9 - struktura systému je znázorněné porovnání programovacích struktur na PC a obecném vestavném systému s mikroprocesorem Vinculum. Ze schématu je pak následně vidět, že u Vinculum II – RTOS můžeme přistupovat k hardwaru jak přes API tak i přes micro-kernel (mikrojádro). Samotná syntaxe příkazů je založena na jazyce C/C++.

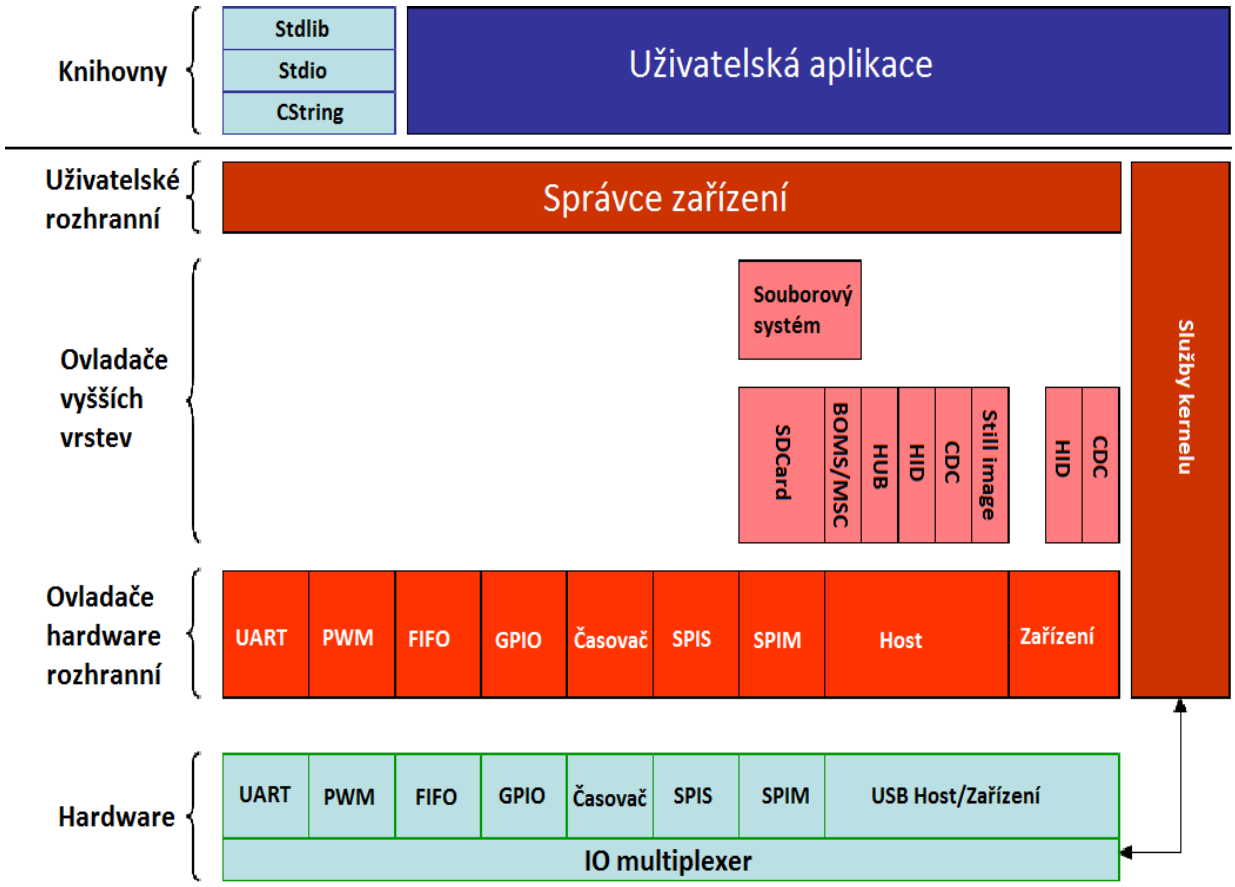


Obrázek 9 - struktura systému

Další výhodu programování nad VOS je také možnost multitaskingu, který je vhodný použít pokud potřebujeme dělat více operací najednou. V případě tohoto programování je nutné znát výrazy jako úloha (task), vlákno (thread) a semafor (v informatice). Poslední výraz jsem uvedl z důvodu, že pomocí tohoto mechanismu je u VOS zajištěna ochrana proti souběhu.

5.1 Architektura softwaru

Na Obrázek 10 - architektura softwaru je znázorněné blokové schéma architektury softwaru pro Vinculum.



Obrázek 10 - architektura softwaru

Při každém RESETU mikroprocesoru se pak musí spustit postupně služby kernelu v následujícím pořadí níže, parametry pro spuštění jednotlivých modulů vyplívají z případné aplikace.

StartVOS: tento funkční blok zinicializuje všechny vnitřní datové struktury a nastaví parametry jádra. Parametrem je kolik ovladačů jádra bude používat, aby mohla být podle toho zorganizována paměť. Dalším parametrem je nastavení časování.

ConfigureIOMux: Jelikož se mikroprocesor vyrábí v mnoha verzích podle počtu pinů, tak parametrem této funkce je verze mikroprocesoru a rozmístění periferií, podle toho se pak multiplexer nastaví. V případě tohoto nastavení je důležité si dávat pozor na rozmístění pinů, které nám umožňují programování obvodu, této nástraze se můžeme vyhnout, když budeme používat vývojové prostředí právě pro procesor Vinculum od výrobce, kde pomocí integrovaných nástrojů si můžeme nastavení multiplexoru předdefinovat a poté se do našeho programu bude samo generovat.

InitializeDriver: Načte veškeré dostupné potřebné ovladače. Optimalizované ovladače pro periferie poskytuje výrobce už ve VOS.

CreateResources: Přidělí zdroje jednotlivým úlohám. V případě, že tyto úlohy (vlákna) spolu interagují, tak jejich kontext se vytváří společně se semafory, mutexy a případně i se sdílenou zásobníkovou pamětí. Každé vlákno má svůj vlastní zásobník, aby mohl VOS sledovat využití paměti.

StartScheduler: Jakmile jsou všechny objekty programu inicializovány, začne běh v reálném čase podle diagramu naplánovaných úloh. VOS používá prioritně založený Round-Robin plánovací algoritmus, tak tedy úlohy s největší prioritou mají přednost. Dále tato funkce sleduje statistiky použití vláken, které nám pak pomáhají při ladění aplikací.

Další ovladače zařízení: Tyto ovladače jsou volitelné podle užitelnosti aplikace. Firma FTDI poskytuje několik těchto ovladačů, které se liší vždy od verze VOS. Mezi tyto ovladače například patří BOMS/MSC (úložná zařízení), HUB, HID (human interface device), SD Card, CDC (např. webkamery) jak je vidět na Obrázek 10 - architektura softwaru. Pro zkušenějšího programátora je také možnost si tyto ovladače vytvořit sám a implementovat do VOS.

Souborový systém: Tato část reprezentuje souborový systém pro použití s přidanými paměťovými zařízeními například flash disky. Od výrobce je dodáván ovladač na souborový systém FAT 12,16 nebo 32. Dále jsou podporovány pouze názvy souboru konvence 8.3, která představuje 8 znaků název souboru a za tečkou 3 znaky přípony. Se soubory se pak po načtení zařízení pracuje přes stejné metody jako v klasickém jazyce C, protože pro tuto možnost je implementována známá knihovna Stdio.

Správce zařízení: Poskytuje rozhranní pro integrované periferie na čipu nebo i jiných zařízení přes patřičné ovladače. Přes toto rozhranní komunikujeme v každém případě, když požadujeme hardware a to platí nejen pro vnitřní programovou strukturu ale i pokud například chceme s hardwarem mikroprocesoru komunikovat přes UART, USB nebo SPI.

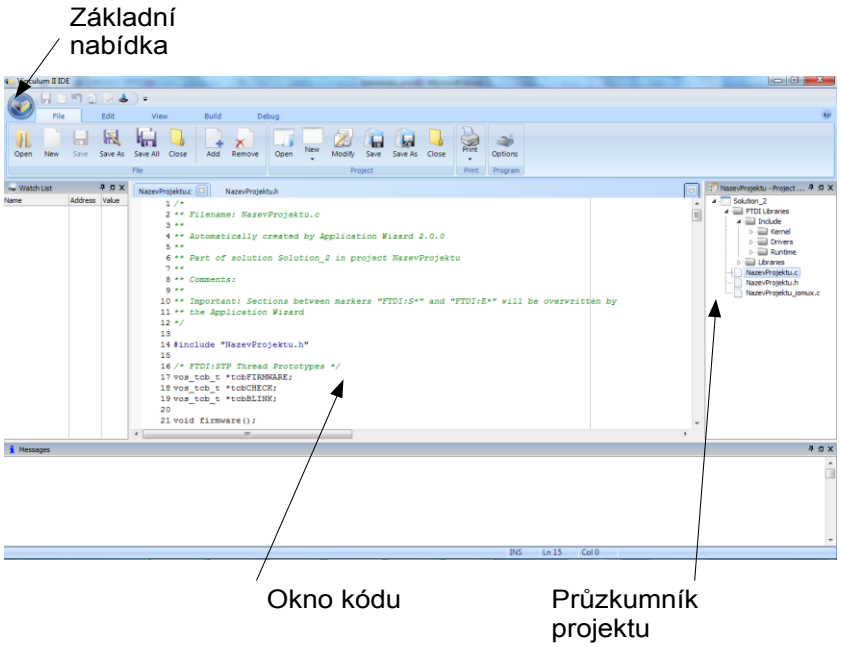
Knihovny: Tato část obsahuje známé knihovny pro práci s řetězci, soubory a spoustu dalších funkcí. Jsou to známé knihovny používané v jazyce C CString, Stdio, Stdlib.

6 Úlohy pro výuku

V této kapitole jsou obsaženy tři jednoduché úlohy koncipovány tak, aby bylo pochopeno sestavení struktury programů pro tento mikroprocesor, jak se mají správně zavádět moduly, pochopení multitaskingu na tomto mikroprocesoru a jeho správa. Jako využité periferie v těchto příkladech pro toto základní pochopení jsem zvolil LED diody, tlačítka na univerzálním kitu a pro práci s flash pamětí USB port na modulu. Jako vhodný materiál k těmto úlohám doporučuji manuál k procesoru, kde v tomto manuálu je seznam všech metod a funkcí nad VOS, jejich návratové hodnoty a syntaxe. K vývoji těchto aplikací je vhodné používat vývojové prostředí od FTDI Vinculum II IDE, které je volně dostupné na webových stránkách FTDI(www.ftdichip.com). Možnosti a základní práci v tomto prostředí popíšu v následující části.

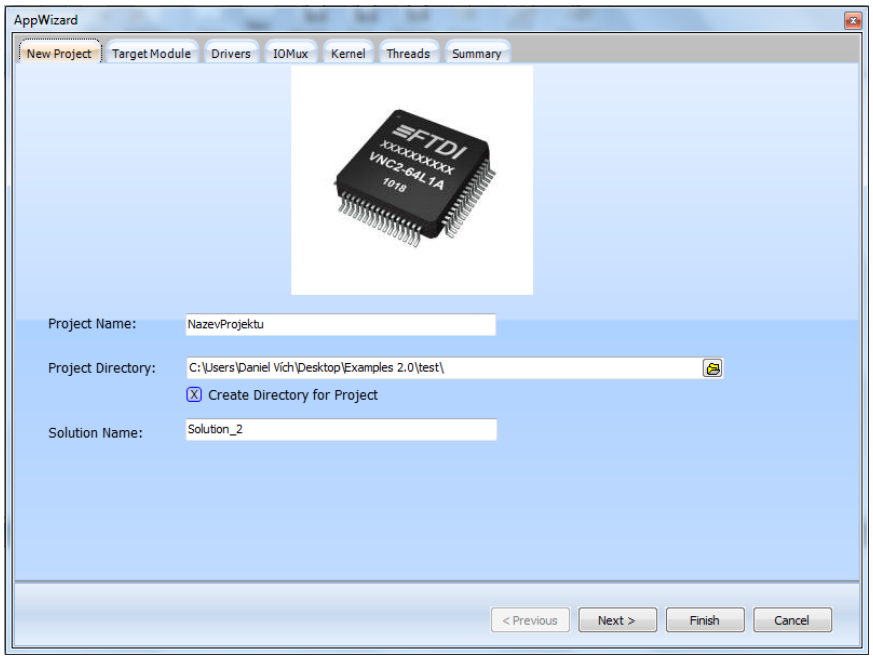
6.1 Vývojové prostředí Vinculum IDE

V této části popíšu základní vytvoření projektu v tomto prostředí pomocí průvodce, kde po zadání údajů o procesoru a potřebných knihovnách pro naši aplikaci se nám vygeneruje inicializační kód, jak jsem výše popisoval. V následujících několika krocích bude vidět nastavení prostředí pro modul s procesorem Vinculum pomocí tohoto průvodce.



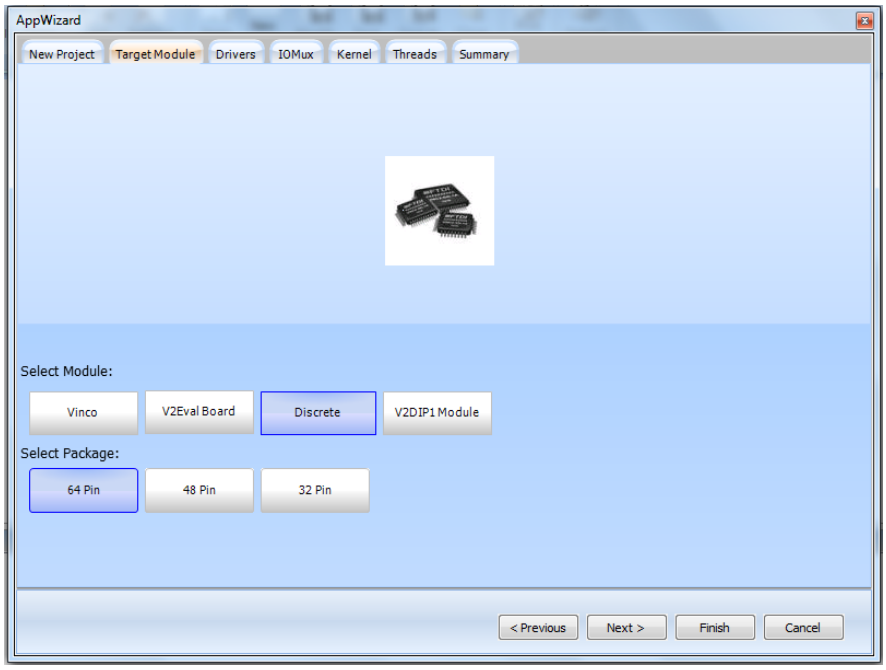
Obrázek 11 - průvodce základní okno

Průvodce spustíme kliknutím na základní nabídku programu volbou *New -> New App Wizard Project.*



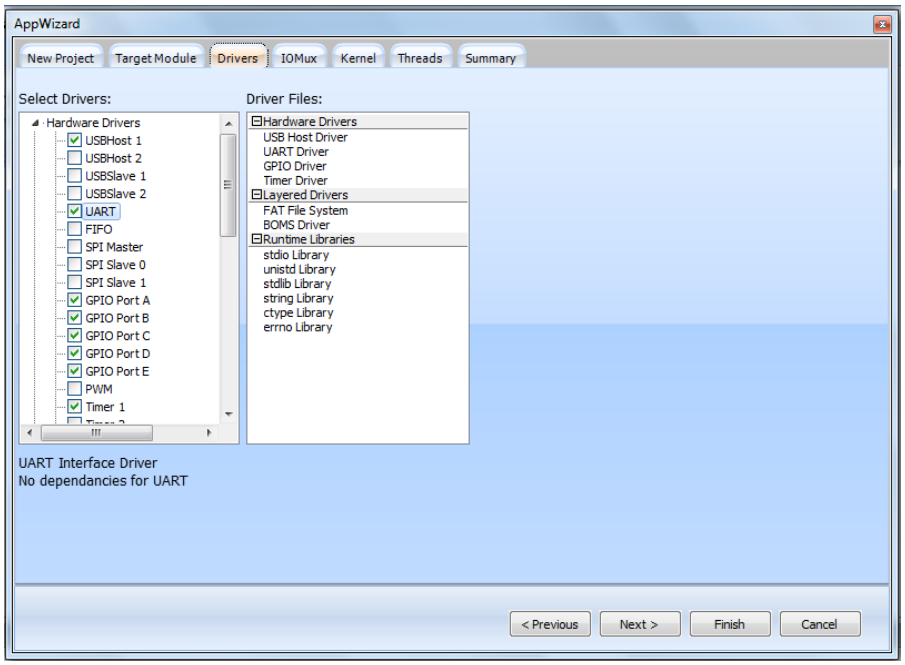
Obrázek 12 - průvodce nastavení projektu

V první části průvodce *New Project* vyplníme pouze název projektu, úložiště a číslo řešení.



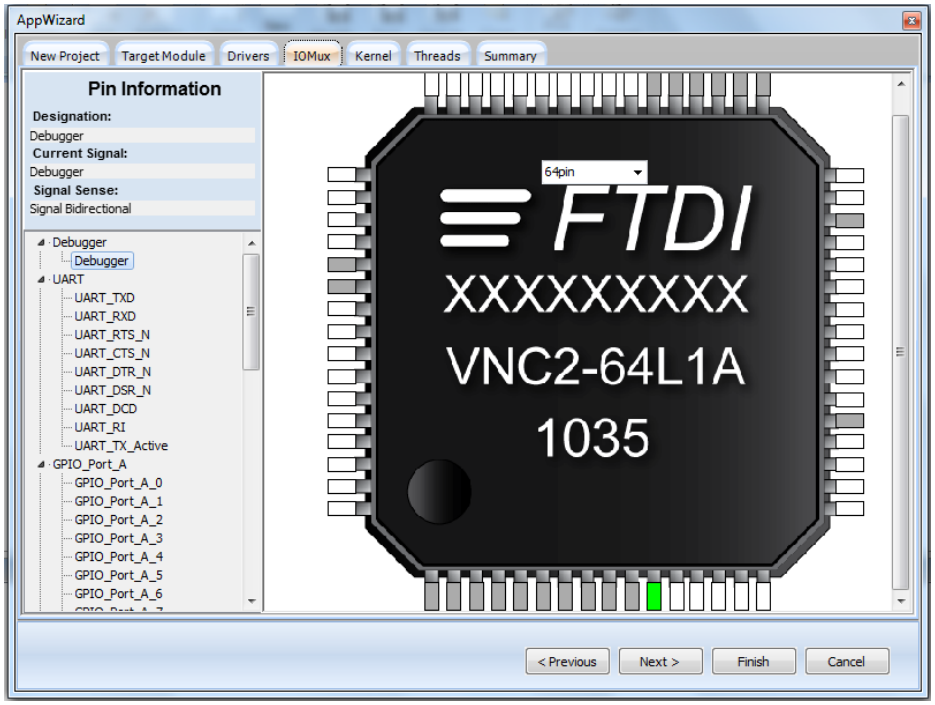
Obrázek 13 - průvodce typ procesoru

V druhé části průvodce *Target Module* je nutné mít zvolené *Discrete* a příslušnou verzi procesoru, v tomto případě *64pin.*



Obrázek 14 - průvodce knihovny

Ve třetí části *Drivers* musíme nastavit požadované ovladače podle vhodnosti pro vytvářenou aplikaci.

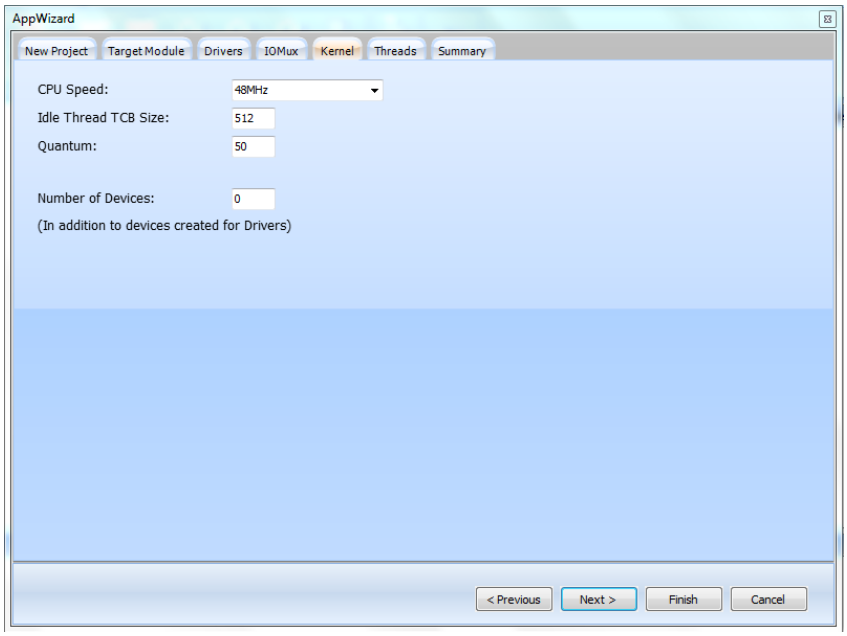


Obrázek 15 - průvodce multiplexor

Ve čtvrté části *IOMux* je důležité nastavení multiplexoru, to musí splňovat konfiguraci, která vyplívá ze schématu zapojení jednotlivých pinů. Zde se také nastavuje nastavení směru komunikace na pinu a jeho další vlastnosti jako volba pull-up nebo pull-down rezistoru. V následující tabulce je seznam rozmístění pinů:

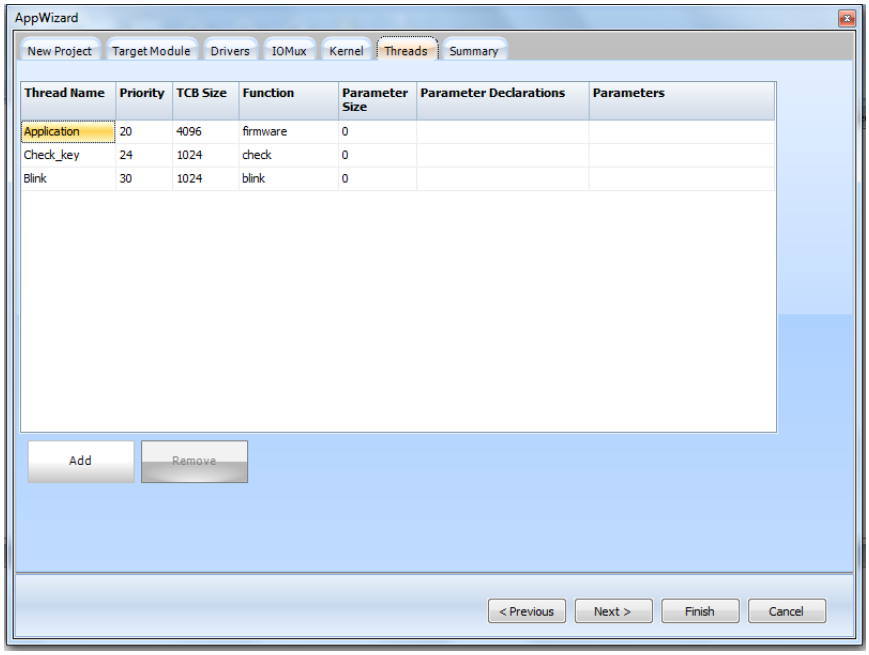
Pin	IOBus	Směr	Typ	Použití
11	0	obousměrný	Debug	Debug rozhranní
12	1	vstup	GPIO_PORT_E	tlačítko 1
13,14,15	2,3,4	obousměrný	GPIO_PORT_D	Rozšiřující port 2
16,17,18	5,6,7	výstup	PWM	Rozšiřující port 1
19,20,22,23, 24,25,26,27	8,9,10,11,12 ,13,14,15	výstup	GPIO_PORT_A	LCD display
28	16	výstup	GPIO_PORT_B	piezo
28	17	obousměrný	GPIO_PORT_B	procesor reálného času
31	18	obousměrný	GPIO_PORT_B	procesor reálného času
32	19	obousměrný	GPIO_PORT_B	DIR485
39	20	výstup	UART	TXD
40	21	vstup	UART	RXD
41	22	výstup	UART	RTS
42	23	vstup	UART	CTS
43,44,45,46, 47	24,25,26,27, 28	výstup	GPIO_PORT_C	Ledbar
48	29	výstup	GPIO_PORT_C	LED 3
49	30	výstup	GPIO_PORT_C	LED 2
50	31	výstup	GPIO_PORT_C	LED 1
51	32	výstup	GPIO_PORT_D	žárovka
52	33	vstup	GPIO_PORT_D	teploměr
55	34	vstup	GPIO_PORT_D	potenciometr
56	35	vstup	GPIO_PORT_D	tlačítko 2
57,58,59,60, 61,62,63,64	36,37,38,39, 40,41,42,43	vstup	GPIO_PORT_E	klávesnice

Tabulka 2 - rozmístění pinů a periferii



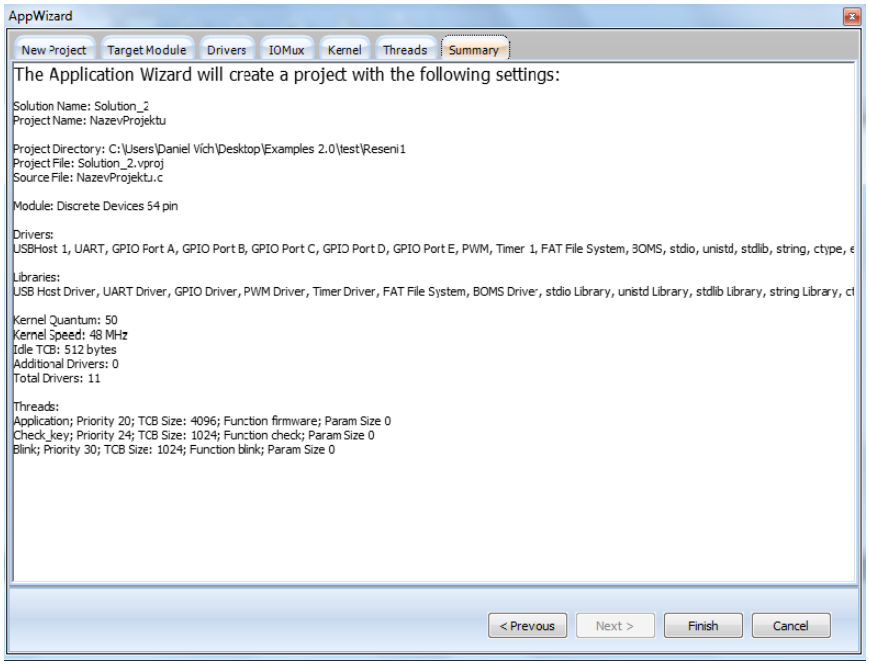
Obrázek 16 - průvodce nastavení kernelu

Zde v páté části se nachází nastavení procesoru jako frekvence, velikost TCB bufferu pro odložené úlohy a počet dalších zařízení. Poslední položka slouží pro přípravu vložení případných ovladačů ostatních zařízení.



Obrázek 17 - průvodce nastavení vláken

V předposlední části se nastavují jednotlivá vlákna programu. Všechna vlákna co zde vytvoříme se automaticky vygenerují, nastaví a vloží do plánovače podle nastavené. Pokud do programu budeme chtít později přidat další vlákno, tak už ho musíme nastavit a přiřadit do plánovače ručně.



Obrázek 18 - průvodce finální zhodnocení

V poslední části se nachází konečný souhrn všech nastavení pro generování kódu. Po kliknutí na tlačítko finish se vygeneruje požadovaný kód. Vlákna co jsme si v průvodci nadefinovali se poté objeví v souboru *NazevProjektu.c* na konci vygenerovaného kódu jako funkce void s názvem, který odpovídá danému vláknu. Do této funkce pak píšeme vlastní program. A plánovač se už postará o běh tohoto programu.

Samozřejmě není nutné používat tohoto průvodce, veškerý kód můžeme napsat ručně. U aplikací s málo periferiemi je to jednodušší a vytvoří se přehlednější kód, ale u větších aplikací je tato volba jednodušší.

6.2 Úloha č. 1 blikající LED dioda

Cílem této úlohy je seznámení se programováním tohoto obvodu, kde jediným úkolem je blikání diody v daném časovém intervalu. Úloha představuje nastavení procesoru, základní inicializaci hardwaru, nastavení pinů, vložení úlohy do plánovače a spuštění plánovače. Veškeré funkce k této úloze jsou dostupné v manuálu k mikroprocesoru. Výsledný program je obsažen na přiloženém cd ve složce uloha_1.

Zadání: Vytvořte program, který bude blikat každou 1 sekundu s jednou z diod na univerzálním kitu.

Postup vypracování: V hlavičkovém souboru si deklarujeme používané globální proměnné a načteme potřebné knihovny. V tomto souboru se také nastaví naše zařízení a načte knihovna vos.h, která zajistí start VOS. V hlavním souboru poté postupujeme s nastavením a inicializací podle výše zmíněného pořadí. Zinicializujeme zařízení, zkontrolujeme typ procesoru a nastavíme multiplexor na požadovaný port:

```
vos_iomux_define_output(29, IOMUX_OUT_GPIO_PORT_C_5);
gpioCtx.port_identifier = GPIO_PORT_C;
 gpio_init(LEDs, &gpioCtx);
```

Dalším krokem je vytvoření funkce pro blikání diody, která bude obsahovat spuštění zařízení, poslání dat na výstup, zpoždění a změnu stavu výstupu.

```
void Blik(void) { BYTE PortData = LED3;
        StartupDevices();
        while (1) {
                vos_delay_msecs(1000);
                PortData ^= LED3;
                  vos_dev_write(hDevice[LED] ,&PortData, 1, NULL);
                }
        }
```

Spuštění této části kódu pak probíhá v plánovači, do kterého tuto část umístíme jako nové vlákno v hlavní metodě spolu s inicializací.

```
vos_create_thread(28, SIZEOF_tcb, &Blik, 0);
```

Nakonec zbývá jen spuštění plánovače úloh.

```
vos_start_scheduler();
```

6.3 Úloha č. 2 blikání 3 diod s použitím plánování úloh

Tato úloha poukazuje na možnost běhu více úloh najednou. V tomto příkladě to budou reprezentovat 3led diody, každá s jinou dobou prodlevy. V této úloze bude použita pro řízení úloh metoda semafor. Inicializace a nastavení je stejné jako v předchozí úloze. Veškeré funkce k této úloze se dají nalézt opět v manuálu k mikroprocesoru. Výsledný program je obsažen na přiloženém cd ve složce uloha_2.

Zadání: Vytvořte program, který bude používat 3 LED diody na univerzálním kitu. Těmto diodám přiřaďte doby prodlev 3s, 2s a 1s. Pro každou diodu vytvořte samostatné vlákno s vlastní dobou prodlevy. Pro synchronizaci použijte mechanismus semafor.

Postup vypracování: Hlavičkový soubor obsahuje stejné informace jako v předchozí úloze. V hlavním souboru jsou však již nějaké menší změny z důvodu použití semaforu, který musíme na začátku programu deklarovat:

```
vos_semaphore_t DevicesStarted;
```

Inicializace zařízení je opět podobná předcházející úloze jen nastavení multiplexoru kvůli použití pinů pro další 2 diody je o tyto 2 prvky delší.

```
vos_iomux_define_output(29, IOMUX_OUT_GPIO_PORT_C_5);
vos_iomux_define_output(30, IOMUX_OUT_GPIO_PORT_C_6);
vos_iomux_define_output(31, IOMUX_OUT_GPIO_PORT_C_7);
gpioCtx.port_identifier = GPIO_PORT_C;
 gpio_init(LEDs, &gpioCtx);
```

Funkce pro jednotlivé diody jsou pak podobné, důležité ale je, aby nedošlou k souběhu těchto úloh, a proto se musí použít semafor. Ten má 2 metody vos_wait_semaphore() a vos_signal_semaphore(). První vos_wait_semaphore() slouží k čekání na uvolnění zdrojů, čítač semaforu je dekrementován a pokud hodnota čítače je menší než nula vlákno je blokováno. Druhá metoda vos_signal_semaphore() provádí operaci signálu na semaforu, kde čítač je inkrementován a pokud je hodnota menší nebo rovna nule tak první vlákno na tomto seznamu je přesunuto do seznamu připravených pro vykonání.

```
void BlikLed1(void) {
  BYTE PortData1 = LED1;
        vos_wait_semaphore(&DevicesStarted);
         vos_signal_semaphore(&DevicesStarted);
  while (1) {
        vos_delay_msecs(1000);
        PortData1 ^= LED1;              // zmena stavu diody
        PortData1 |=last_data ;         //zmena minuleho stavu pouze na teto diod

        // poslani hodnoty v promenné portdata na vystup
        vos_dev_write(hDevice[LEDs] ,&PortData1, 1, NULL);
    }
  }
```

V hlavní metodě mimo nastavení jako v předchozí úloze zbývá semafor inicializovat, vytvořit jednotlivá vlákna a spustit plánovač.

```
vos_init_semaphore(&DevicesStarted, 0);
vos_create_thread(27, SIZEOF_tcb, &BlikLed1, 0);
 vos_create_thread(28, SIZEOF_tcb, &BlikLed2, 0);
 vos_create_thread(29, SIZEOF_tcb, &BlikLed3, 0);
 vos_start_scheduler();
```

6.4 Úloha č. 3 tlačítka a záznam dat na USB flash disk

Tato úloha poukazuje na práci se vstupním zařízením v tomto případě tlačítka. Dále jednoduchost zápisu dat na USB flash disk, kde v tomto případě se bude do textového souboru na flash disku zapisovat počet stisknutí tlačítek. Opět se k řízení úloh budou používat semafory. Výsledný program je obsažen na přiloženém cd ve složce uloha_3.

Zadání: Přidejte do úlohy č. 2 podporu dvou tlačítek opět z univerzálního kitu, která budou mít funkci zvýšení a snížení doby prodlevy blikání všech 3 led diod o 100ms. Dále zapisujte do textového souboru klik.txt na USB flash disk počet zmáčknutí tlačítek.

Postup vypracování: Začíná opět stejně jako v předchozích úlohách nadefinováním hlavičkového souboru, kde se nacházejí potřebné definice a deklarace. Postup vytváření hlavního souboru je také stejný, jako v předchozích úlohách jen opět přibyli další zařízení tlačítka a USB flash disk, takže nám přibudou další inicializace a nastavení zařízení. Pro přístup k flash disku potřebujeme inicializovat USB host, BOMS (mass storage device) a implementaci souborového systému FAT:

```
usb_ctx.if_count = 1;
usbhost_init(-1, VOS_DEV_USB_HOST, &usb_ctx);
boms_init(VOS_DEV_BOMS);
fat_init();
```

Dále budeme potřebovat pro tlačítka mít nastavené správné piny procesoru na vstup.

```
vos_iomux_define_input(12, IOMUX_IN_GPIO_PORT_E_5);
vos_iomux_define_input(56, IOMUX_IN_GPIO_PORT_D_7);
gpioCtx.port_identifier = GPIO_PORT_E;
gpio_init(Button1, &gpioCtx);
gpioCtx.port_identifier = GPIO_PORT_D;
 gpio_init(Button2, &gpioCtx);
```

Dalším krokem je připojení a konfigurace USB, přiřazení třídy BOMS k USB a připojení souborového systému FAT.

```
//pripojeni usb
hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_GET_CONNECT_STATE;
vos_dev_ioctl(hUsb, &hc_iocb);

// najdeme mass storage (flash disk)
hc_iocb_class.dev_class = USB_CLASS_MASS_STORAGE;
hc_iocb_class.dev_subclass = USB_SUBCLASS_MASS_STORAGE_SCSI;
hc_iocb_class.dev_protocol = USB_PROTOCOL_MASS_STORAGE_BOMS;
hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_DEVICE_FIND_HANDLE_BY_CLASS;
hc_iocb.handle.dif = NULL;
hc_iocb.set = &hc_iocb_class;
hc_iocb.get = &ifDev;
vos_dev_ioctl(hUsb, &hc_iocb);

//pripojime k flash disku podporu mass storage
boms_att.hc_handle = hUsb;
boms_iocb.ioctl_code = MSI_IOCTL_BOMS_ATTACH;
boms_iocb.set = &boms_att;
boms_iocb.get = NULL;

//pripojeni FAT
fat_ioctl.ioctl_code = FAT_IOCTL_FS_ATTACH;
fat_ioctl.set = &fat_att;
fat_att.partition = 0;
fsAttach(hFat);
```

Funkce pro blikání diod jsou stejné, jako u předchozího příkladu jen hodnoty jednotlivých zpoždění předáváme pomocí proměnných Delay1, Delay2 a Delay3. Tyto proměnné pak při stisku příslušného tlačítka dekrementujeme nebo inkrementujeme. Funkce pro rozpoznání

stisku tlačítek pak obsahuje i zápis do souboru pomocí metod známé knihovny stdio a vypadá následovně:

```
void Tlac1() {
   gpio_ioctl_cb_t gpio_iocb;
   vos_wait_semaphore(&DevicesStarted);
   vos_signal_semaphore(&DevicesStarted);
    gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_SET_PROG_INT0_PIN;
   gpio_iocb.value = GPIO_PIN_5;
   vos_dev_ioctl(hDevice[Button1], &gpio_iocb);
   while (1) {
         // Cekani na zmacknuti tlacitka
          gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_SET_PROG_INT0_MODE;
         gpio_iocb.value = GPIO_INT_ON_NEG_EDGE;
         vos_dev_ioctl(hDevice[Button1], &gpio_iocb);
         gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_WAIT_ON_INT0;
          vos_dev_ioctl(hDevice[Button1], &gpio_iocb);
         // Tlacitko stisknuto
         Delay1 += 100;
          Delay2 += 100;
          Delay3 += 100;

         //zapis do souboru
         file = fopen("klik.txt", "w");
         counter++;
         fprintf(file, "aktualni stav stisku: %d", counter);
         fclose(file);
      }
   }
```

Funkce výše je pro inkrementační tlačítko, pro dekrementační vypadá podobně jen po stisknutí tlačítka hodnoty zpoždění Delay dekrementujeme, vhodné je pak mít ošetřené, aby při dosažení nuly dekrementace neprobíhala.

```
if (Delay1 != 0){Delay1 -= 100;}
if (Delay2 != 0){Delay2 -= 100;}
if (Delay3 != 0){Delay3 -= 100;}
```

Nakonec opět zbývá jen vytvořit vlákna a spustit plánovač.

```
vos_create_thread(25, SIZEOF_tcb, &Tlac1, 0);
vos_create_thread(26, SIZEOF_tcb, &Tlac2, 0);
vos_create_thread(27, SIZEOF_tcb, &BlikLed1, 0);
 vos_create_thread(28, SIZEOF_tcb, &BlikLed2, 0);
 vos_create_thread(29, SIZEOF_tcb, &BlikLed3, 0);
vos_start_scheduler();
```

Závěr

Tato práce měla za cíl seznámení se s obvody Vinculum od firmy FTDI, jejich možnostmi, následné vytvoření návrhu zapojení modulu, který bude rozšiřovat již hotový univerzální výukový přípravek, realizaci tohoto zapojení a vytvoření sady příkladů pro výuku demonstrující možnosti toho mikroprocesoru. I když během zpracování práce se vyskytovali nějaké problémy při návrhu, zejména při rozmístění příslušných konektorů, jak je zmiňováno výše, byly všechny body této práce úspěšně splněny.

V této práci také poukazuji na možnosti programování tohoto mikroprocesoru Vinculum pomocí výrobcem integrovaného operačního systému, který nám usnadní práci nad hardwarem a řeší za nás některé složité úkoly jako je například multitasking, přidělování zdrojů, přerušení, komunikace s USB zařízeními a mnoho dalších.

Tři návrhy úloh určených pro výuku mají poukazovat na základní práci s tímto modulem, konkrétně ovládání výstupu, získávání dat ze vstupu, komunikaci po USB a multitasking. Samozřejmě, že tyto úlohy nevyužijí naplno možnosti mikroprocesoru Vinculum, ale dají po těchto znalostech základní impuls k tvorbě složitějších aplikací, které jsou už jen nabalováním dalších funkcí na sebe.

Co se týče zhodnocení vlastností mikroprocesoru, tak ohledně výkonu a velikostí paměti nevybočuje z řady mnoha jiných podobné struktury, ale výhodou oproti ostatním je integrace rozhranní USB a dále podpora od výrobce ohledně vývoje softwaru pro tyto mikroprocesory. Konkrétně můžu zmínit již několikrát popisovaný VOS (Vinculum operating system), velké množství již hotových knihoven pro různé aplikace a rozhranní a také vlastní vývojové prostředí s plnou podporou k tomuto mikroprocesoru.

Můj osobní přínos z této práce je, že jsem si opět více osvojil návrh zapojení obvodů, jeho realizaci a také možnosti právě mikroprocesorů Vinculum pro případné další použití například v nějakých vestavných systémech pro určitý typ aplikace. Další výhodnou znalostí je programování těchto obvodů a to včetně přidaných USB zařízení.

Použité zdroje

[1] *VINCULUM-II EMBEDDED DUAL USB HOST CONTROLLER IC Datasheet* [online]. [s.l.] : [s.n.], 2010 [cit.2011-05-31]. Dostupné z WWW: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_Vinculum-II.pdf>

[2] *Vinculum VNC1L Embedded USB Host Controller IC Datasheet Version 2.02* [online]. [s.l.] : [s.n.], 2009 [cit.2011-05-31]. Dostupné z WWW: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_VNC1L.pdf>

[3] *Vinculum Firmware User Manual Version 2.05* [online]. 373 Scotland Street, Glasgow G5 8QB United Kingdom : [s.n.], 2008 [cit. 2011-05-31]. Dostupné z WWW: <http://www.ftdichip.com/Firmware/Precompiled/UM_VinculumFirmware_V205.pdf>

[4] *Vinco Development Module Datasheet Version 2.0* [online]. [s.l.] : [s.n.], 2011 [cit. 2011-05-31]. Dostupné z WWW: <http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_Vinco.pdf>

[5] *OpenSourceProject.org.cn* [online]. 2007 [cit. 2011-05-31]. The Scheduler. Dostupné z WWW: <http://book.opensourceproject.org.cn/embedded/cmprealtime/opensource/5107final/lib0024.html>

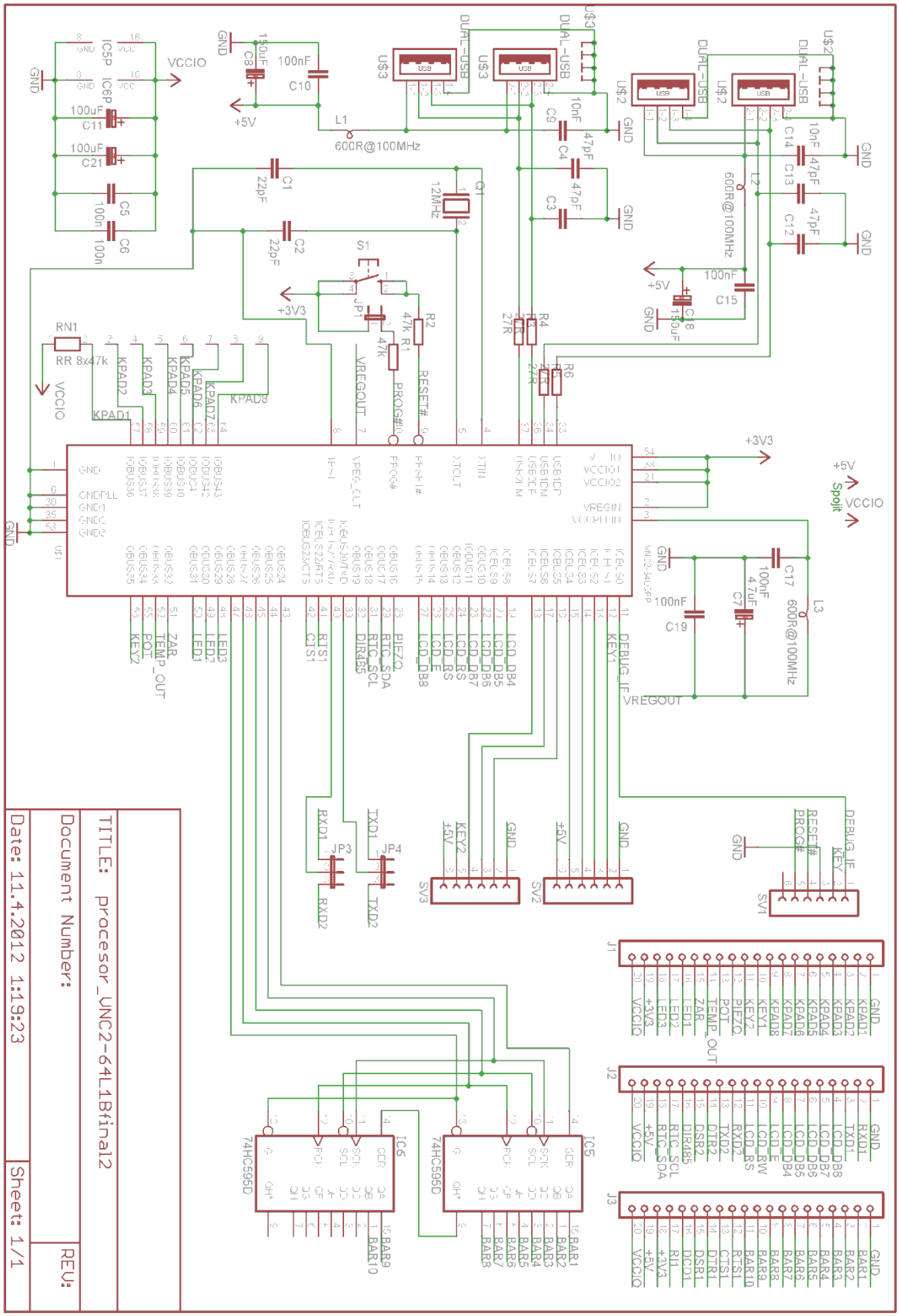
[6] Vinculum. In Pandatron. [online]. [s.l.] : [s.n.], 2007 [cit. 2011-05-31]. Dostupné z WWW: <http://pandatron.cz/?158&vinculum_dil_1.>

[7] *Ghielectronics.com* [online]. 2011 [cit. 2011-05-31]. Dostupné z WWW: <http://www.ghielectronics.com>

[8] *Soselectronic.cz* [online]. 2010 [cit. 2011-05-31]. Vinculum-II - druhá generace USB Host / Slave řadičů. Dostupné z WWW: <http://www.soselectronic.cz/?str=806>

[9] HYDE, John. Embedded USB design by examples [online]. 2010 [cit. 2012-05-17]. Dostupné z: <http://www.scribd.com/doc/46457539/EUDBE2-01>

Příloha A


TITLE: procesor_UNC2-64L1Bfinal2
Document Number:
REV:
Date: 11.4.2012 1:19:23
Sheet: 1/1

Příloha B


Vyukovy pripravek - procesor VNC2-64L1B
Verze 1.1
by Daniel Vich
IC5
IC6
PROG
SET
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D